SIACG 2002 - 1st Ibero-American Symposium on Computer Graphics
1-5 July 2002, Guimarães - Portugal

3D Texture-based Hybrid Visualizations

Imma Boada
Institut Informàtica i Aplicacions, IIiA
Universitat de Girona
Girona
imma@ima.udg.es

Isabel Navazo
Dept. Llenguatges i Sistemes Informàtics
Univ. Politècnica de Catalunya
Barcelona
isabel@lsi.upc.es

Abstract

In this paper the visualization of hybrid scenes that contain volume data and a fitted extracted surface is addressed. The proposed algorithm is based on a integrated octree-based representation: the "Hybrid Octree". The Hybrid Octree allows to obtain multiresolution representation of the volume data and it also maintains a decimated surface codification. The proposed visualization approach uses 3D-textures for the visualization of the volume data and integrates the surface polygons using the information represented in the octree structure. The main characteristics of the method are: its capabilities to perform multiresolution hybrid visualizations and its efficient use of texture memory space.

Keywords

Hybrid Visualization, Octree data structures, 3D textures.

1 INTRODUCTION

The incorporation of volumetric and geometric objects, either synthetic (i.e. modeled with CAD tools) or fitted (i.e. extracted from volume data), into one scene is a process referred to as *Hybrid Modeling*. Hybrid Modeling plays an important role in a large number of scientific visualization applications.

Two main approaches have been proposed to deal with hybrid scenes. The first reduces surface and volume data to a common codification scheme and applies classical surface and volume data visualization strategies [9, 10, 11, 21, 25]. The second maintains surface and volume data in their original representation schemes and defines specialized renderers able to combine all the data. In this last case the data integration step is part of the visualization process which requires surface and volume data to be composed in depth sorted order [15, 24, 12, 26, 13].

Following the second strategy we present a novel 3D texture-based hybrid visualization function able to combine surface and volume data. The proposed visualization function exploits the 3D texture graphics hardware capabilities and it is able to perform multiresolution hybrid visualizations which guarantee an efficient use of the texture memory space. The two key points of this visualization function are:

- *The data structure used to represent surface and volume data.* The Hybrid Octree (HO), a hierarchical data structure based on an octree representation that maintains surface and volume data simultaneously and implicitly ordered [2].
- *The policy applied to represent volume data in texture memory.* To perform hybrid visualization we propose an extension of the multiresolution texture memory assignation policy presented in [4]. This strategy uses a compressed representation of homogeneous regions and no importance areas of the volume data in order to reduce the texture memory space required to maintain the data and therefore, improve rendering speed.

The paper is structured as follows. Section 2 reviews the HO, the data structure used to maintain the hybrid scene, giving a description of its construction process. Section 3 describes the main considerations that have to be taken into account to deal with hybrid scenes. The proposed visualization function is presented in Section 4. Section 5 presents and discusses the results on several practical examples. Finally, Conclusions and Future Work are given in Section 6.

2 OCTREE BASED CODIFICATIONS

The octree model, originally introduced for solid representations [17, 22, 20], is a tree that codes the recursive subdivision of a finite cubic universe. The root of the tree represents the universe, a cube with 2^n edge length. This cube is divided into eight identical cubes, called octants, with an edge length of 2^{n-1}. Each octant is represented by one of

the eight descendants of the root. This subdivision process is repeated recursively until octants contain data that can be represented exactly (named Terminal Octree Nodes) or octants have a minimum edge length (Minimal Resolution Nodes).

Since the earliest work from Meagher [17] concerning 3D data representation based on the octree scheme, many applications of this hierarchical representation have appeared over the years, e.g., to improve rendering speed, to manage easily large data sets, or to obtain error bounded renderings [15, 14, 29, 30]. In surface rendering applications, they have been used in combination with the Marching Cubes (MC) algorithm [16] for accelerating the surface extraction process or as a *surface simplification method* for adaptive isosurface extraction [23, 27].

In our previous work we focused our interest in how octree data structures can be exploited to maintain surface and volume data. In particular we propose the three octree-based data structures presented in the following sections.

2.1 The Volume Octree.

The Volume Octree[4], (VO), is an octree-based data structure used to code homogeneous regions of 3D regular sampled data in a compressed way. For each VO node we maintain information of the maximum error which is introduced if each sample data inside its associated octant is approximated by a trilinear interpolation of the eight values represented on the corners of the octant. This error, denoted the *nodal error* (ε_0), is used to obtain multiresolution volume representations by applying error-driven adaptive traversals.

Being ϵ_u the user required degree of accuracy VO nodes for which their nodal error $\varepsilon_0 \leq \epsilon_u$ are selected and used to define a multiresolution representations of the volume in texture space. This compressed representation reduces the space of texture memory required to obtain 3D texture-based volume data visualizations.

2.2 The Surface Octree.

The Surface Octree [5], (SO) is an octree-based codification used to maintain a decimated codification of a fitted surface obtained by the Discretized Marching Cubes algorithm (DiscMC) [18, 19]. Additionally to the classical black, white and grey nodes [22], the SO introduces five new terminal nodes able to codify any DiscMC surface(see figure 1).

The SO codification reduces the number of faces of the fitted surface without introducing any error [5] and provides a framework able to support multiresolution surface reconstructions [6].

2.3 The Hybrid Octree

The HO presented in [3] is the integration of a VO and a SO. The HO construction algorithm is based on the following steps:

1. *Volume Data Integration* This phase starts with the construction of a VO. For each node n_i of the octree we maintain: (i) the maximum max and minimum min values of the region covered by the node (they are used to improve the surface extraction process required in the next phase [29]); (ii) a list of *nodal errors* (see subsection 2.1.) that represents the degree of homogeneity of the area covered by the node.

2. *Surface Data Integration* This phase detects the terminal nodes intersected by the surface (terminal nodes such that $max \leq isovalue \leq min$, where $isovalue$ identifies the surface) and codifies them with the corresponding Terminal Surface pattern (see figure 1). Subsequently, a surface compression process is applied to reduce the number of faces. To guarantee the continuity of the surface several tests are carried out during the compression process.

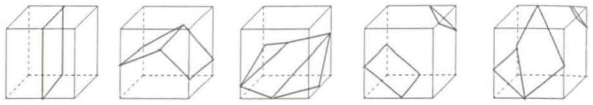

Figure 1. Terminal Surface nodes added to the classical octree.

At the end of this construction process the HO maintains an exact representation of a surface and an error-based volume data codification.

Note that given a node of the HO we are able to identify all the surface polygons contained in this node and we are able to represent the volume data of this node with an ϵ_u degree of accuracy. Moreover, the hierarchical nature of the HO facilitates the generation of multiresolution hybrid scenes just selecting HO nodes distributed at different levels of the HO. For the visualization of these hybrid scenes we propose the 3D texture-based hybrid visualization function presented in Section 4.

3 HYBRID VISUALIZATIONS

Several approaches have been proposed to combine surface and volume data in a single image, based on an image order ray-tracing [15, 24] or on a projective approach [12, 26, 13]. The critical point of all of them falls on the composition process that is required to visualize the data in the correct depth sorted order. A detailed description of this process is given in next section.

3.1 Volume and Surface data Integration

Let us consider the situation represented in figure 2(a) as an example to describe the problem behind the integration of surface and volume data on a single image. In this illustration a scene composed of a volume model (the white ellipse) and a surface (the dashed line) are casted by a ray. In order to determine the color and opacity values of the final image pixel, independent samples of surface and volume data have to be combined 2(b)(c). The accumulated opacity and color can be expressed as the back-to-front recursive formulae [15]:

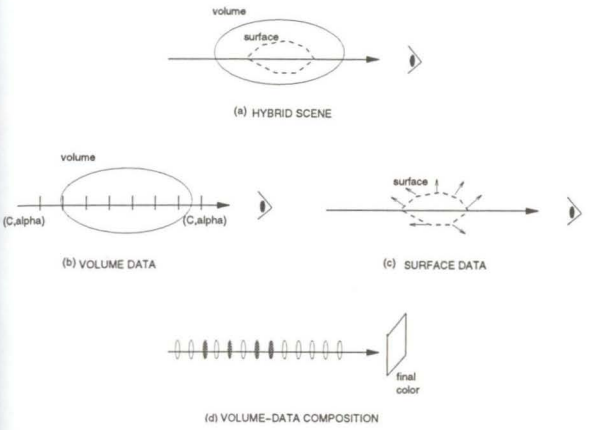

Figure 2. Volume and Surface data are traversed by a ray.

$$C_{out} = (1 - \alpha_{in})C_i + C_{in}$$

and

$$\alpha_{out} = (1 - \alpha_{in})\alpha_i + \alpha_{in}$$

where C_i and α_i values are substituted by volume data or surface data color and opacity values. If the i contribution corresponds to volume data, then C_i and α_i are substituted by the color and opacity values assigned to the voxel sample. If contribution i represents the ray polygon intersection point, then they are substituted by the color and opacity values assigned to this point. Once all surface and volume data have been composited the final C_{out} and α_{out} represent the color and opacity values assigned to the pixel 2(d).

3.2 3D Texture-based hybrid visualization

Let us consider that instead of that hybrid ray-tracer we apply a 3D texture-based hybrid visualizer. In this case volume data has to be rendered by the back-to-front composition of a set of polygons that slices and samples the volumetric dataset which is loaded into the texture memory of the graphics subsystem (see figure 3(a)). In order to integrate the surface data, the surface polygons have to be properly composited between the textured slices (see figure 3(b)).

Two situations have to be considered:

- *The hybrid scene is composed of opaque surfaces.*

  In this case the hybrid rendering is straightforward. Opaque polygons are drawn first taking enabled the depth buffer test. Then the volume samples either opaque or translucent are rendered from back-to-front according to view-point position. This last step is done using 3D textures.

- *The hybrid scene is composed of transparent surfaces.*

  In this case the surface polygons have to be rendered and composited in between the textured slices,

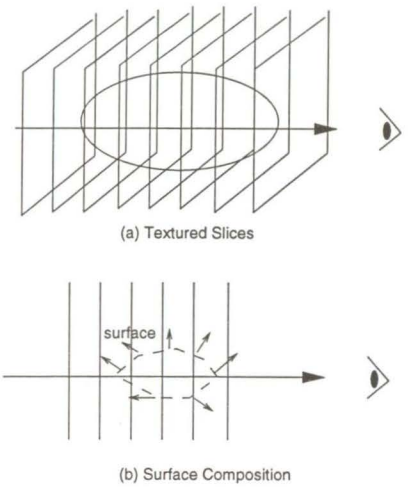

Figure 3. (a)Original Volume data and its 3D texture definition. (b) Back-to-front composition of textured slices.

(*slabs*), in the correct order. Two different alternatives have been proposed by Kreeger et al.[13] to perform such a integration process:

- The first alternative performs a clipping process for each slab. To process the $slab_i$ (i.e. the space between $slice_{i-1}$ and $slice_i$) two clipping planes are enabled. The first is located on the $slice_{i-1}$ and the second on the $slice_i$. Then all surface polygons are sent to the graphics pipeline. This process guarantees that only portions of the polygons contained in the $slab_i$ are rendered. The load of the polygon pipeline increases considerably.
- The second alternative defines additional data structures (such as buckets, active triangle list, ...) which maintains for each slab the list of polygons contained in it.

Our proposed algorithm exploits the information represented in HO nodes and determines the correct position of surface polygons just analyzing the information of each octree node. This strategy simplifies the sorting of surface polygons between the textured slices. Moreover, the proposed alternative has two additional advantages. By one hand, as the HO maintains a multiresolution representation of the volume data (encoded as a list of nodal errors), we can apply the multiresolution texture memory representation policy presented in [4]. Therefore, we can guarantee an optimal use of texture memory space with a compressed representation of homogeneous regions and no importance areas of the volume.

On the other hand, the correct location of the surface polygons is independent of the degree of accuracy at which volume data is represented. Therefore, the sorting process can be applied to obtain multiresolution hybrid visualizations.

4 THE HYBRID OCTREE VISUALIZATION

Note that the HO visualization is equivalent to the hybrid visualization of a set of HO nodes. Hence once the hybrid rendering of a node has been solved it is enough to define a method to determine the set of nodes from which the final image will be obtained. For this reason our work has been centered on the visualization of a HO node.

4.1 The HO Node Visualization Function

Let us considered that the visualization function is applied to a n_i node of the HO, which is intersected by the surface (but it is not a Terminal node). This function performs the three following steps:

- **Texture Generation**

  The volume information covered by the n_i node is represented in texture memory (see figure4(a)). Two different strategies can be applied:

  - The classical assignment policy of one texel per voxel [1, 8, 7, 28]. In this case, the values in the texture memory correspond to original volume data values covered by the node.
  - A more synthetical representation in which one texel can represent multiple voxels values. This is the policy we are going to apply, giving a more compact representation to the given node, either because the node corresponds to a nearly homogeneous region, or because it is contained in a region of low importance, and thus it can be represented in a compressed manner.

    Note that for each node n_i at level k we have a corresponding subtree of depth $l_{max} - k$ and a corresponding voxel region of resolution $2^{l_{max}-k} * 2^{l_{max}-k} * 2^{l_{max}-k}$. Therefore, we can represent the voxel region associated with n_i by using any one of the regularly sub-sampled representations built on the initial voxel values. If we select a voxel set at the deepest level we end up with the case described above (one texel for each voxel). In all other cases, we select a more compact representation corresponding to nodes which are p levels below in the HO structure (with $p = 0 .. l_{max} - k$). The *texture memory* required to represent this node is $2^p * 2^p * 2^p$. The number of slices (ns) used to render the node is set according to the resolution of the defined texture, in particular we define $ns = 2^p$ (see figure4(b)).

- **Integration of Surface Polygons**

  Once the texture has been defined surface polygons are integrated between the texture slices. Two considerations have to be taken into account to perform this integration process:

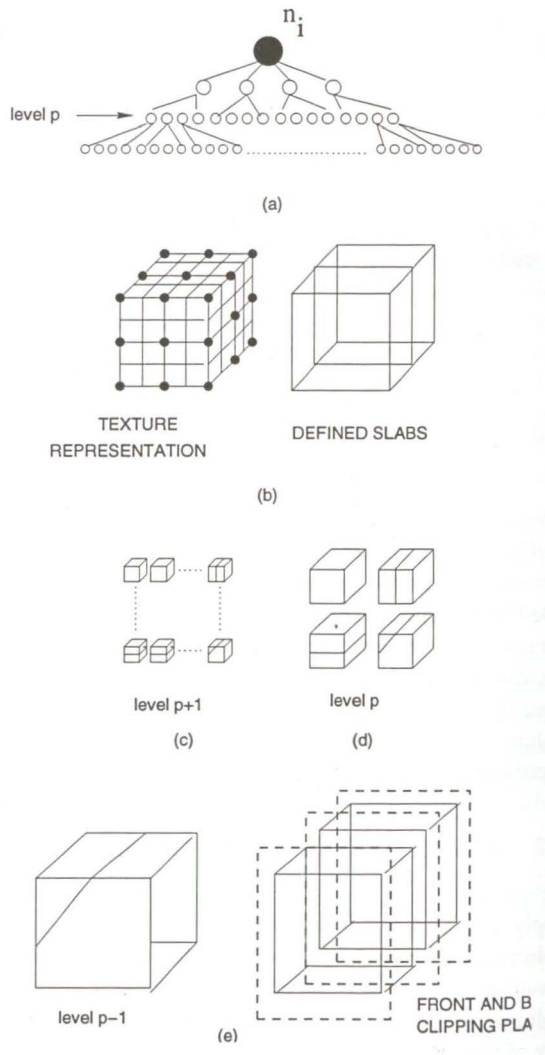

Figure 4. (a) The subtree rooted in n_i (b) The texture represents volume data obtained from level p. (c)(d) If surface nodes are from level p or $p+1$ only one slab is intersected. (e) Surface nodes of level $p-1$ intersect more than one slab. In this case the activation of front and back clipping planes guarantees that only surface fragments contained in the slab are rendered.

- *The slab definition is completely dependent of how volume information contained in the node is represented in the texture memory.*

  Following the description presented in the previous step we know that: (i) the number of slices (ns) used to render a node is set according to the resolution of the defined texture, in our case $ns = 2^p$; (ii) each slice corresponds to a plane of $2^{l_{max}-k}$x$2^{l_{max}-k}$ parallel to the xy face; (iii) one slab is defined as the volume compressed between two consecutive slices. Thus, if 2^p is the number of slices the number of node's slabs is 2^{p-1}; (iv) if e is the size of maximal subdivision nodes, the slab corresponds to the parallelipidical volume of $2^{l_{max}-k}$ x $2^{l_{max}-k}$ x $(e * \frac{2^{l_{max}-k}}{2^p})$. The *origin of the slab* is the origin of the first slice that defines it and the *final extreme* is the maximal vertex of the second slice that define it. The *width of the slab* , w, is $e * \frac{2^{l_{max}-k}}{2^p}$.

- *The surface reconstruction process has to be applied at each one of the nodes of the subtree rooted by n_i intersected by the surface [5].*

  To carry out a hybrid visualization it is necessary to know the slab in which each surface polygon has to be rendered. As the HO maintains the connection between the surface and the volume data, during the surface reconstruction process we introduce a *slab integration* phase. This phase generates the *SlabPolygon* list which has one entry for each intersected slab and maintains for each entry the set of surface nodes contained in this slab.

  Consider, for example, n_s as one of the selected surface nodes where (o_x, o_y, o_z) and l are the origin and the level of this node. The slab integration process compares l with the level selected to represent volume data in the texture (level p in our case):

  (i) If $l \geq p$ the size of the surface node's edge is lower or equal to the slab's width. Therefore, the polygons of the terminal surface node only intersect one slab (see figure 4(c)(d)). If (x_0, y_0, z_0) is the origin of the n_i node and the slabs are labeled from 0 to 2^{p-1}, being slab 0 the slab of origin (x_0, y_0, z_0) we obtain the number of the intersected slab as $\frac{o_z - z_0}{w}$. This parameter determines the entries of the SlabPolygons list to which the node has to be added.

  (ii) If $l < p$ the slab's width is less than the size of the node's edge. The polygons represented in the node intersect more than one slab (see figure 4(e)). The number of intersected slabs is obtained from: $e * 2^l$ The node intersects slabs from $\frac{o_z - z_0}{w}$ to $\frac{o_z - z_0}{w} + e * 2^l$.

- **Visualization**

  Finally when the texture T_i has been defined and the *SlabPolygons* list has been constructed we obtain the visualization of the node n_i applying the Node Visualization function. This function applies the following steps:

  1. BOUNDING CUBE DEFINITION. A bounding cube centered on the center of the node n_i is defined. The ns polygons to be rendered form series of slices through this bounding node, parallel to the xy face (see figure 5(a) and (b)).
  2. CLIPPING PLANES ACTIVATION. The set of ns polygons to be rendered always remains parallel to the projection plane and extended to the bounding cube. The node's volume is represented by T and rotates into the bounding node according to the viewing direction (see Figure 5(b) and (d)). To guarantee that only the volume information represented in the texture will contribute to the final image (i.e. external areas of the node contained in the bounding node are eliminated) a set of clipping planes is defined. The clipping planes are positioned according to the viewing direction at each one of the node's volume faces (see Figure 5(c) and (e)). Lateral, up and down clipping planes are set only once, while the back and front clipping planes should be enabled and disabled more than once. To understand why the latter can be enabled/disabled multiple times, consider the situation in which one polygon intersects more than one slab (see figure 4(c)).
  3. SLAB POLYGONS POSITIONING. This step has to be applied each time the view position changes. The SlabPolygons list has been defined considering the node in its original orientation. Thus, intersected surface nodes are correctly positioned in between the slabs and consequently the stored order guarantees the rendering in the correct slab.

     When the viewer position is modified the 3D textured polygons are maintained parallel to the projection plane. The texture is rotated according to the new orientation and clipping planes are positioned according to the new viewing direction so that each one is in one of the node's faces (lateral, up and down). The rotation has also to be applied on the surface polygons, which in the new position might fall in a different slab. Thus, it is necessary to update the SlabPolygons list to maintain the correct situation. The correct assignation of the polygons within the slabs is performed by the UpdatePosition function. This function determines the new slab position by applying the viewing transformation to the z component of the origin of the surface node.

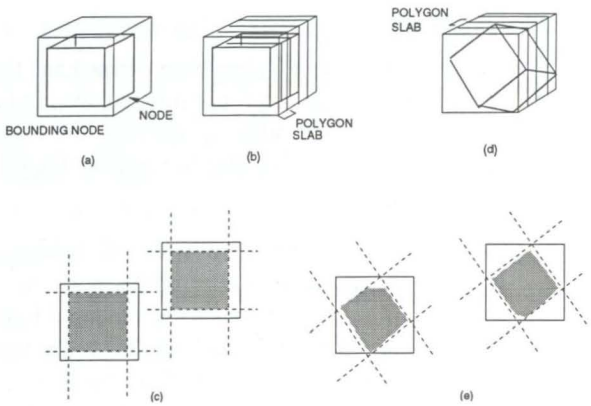

Figure 5. (a) The bounding node's definition. (b) Polygons slices. (c) Activation of the clipping planes. (d) a new orientation requires a rotation of texture space. (e) clipping planes are positioned according the new orientation.

4. NODE'S POSITIONING. The next step is to determine where the volume and surface polygons have to be projected. The projection position is fixed by the node's octree position and by the viewing direction. These parameters determine the current set of geometrical transformations to be carried out.
5. RENDERING Finally the clipped textured polygons and the surface polygons are composited to generate the final image.

4.2 Importance driven Hybrid Visualizations

Once the rendering of a HO node has been solved we propose an importance driven hybrid visualization algorithm. This algorithm is composed of two steps:

1. The first step determines which nodes of the HO have to be rendered. Once the HO has been constructed a region of maximal interest is selected. Then restricted by the texture memory capacity we apply a node's selection process that determines the nodes that have to be used for the visualization. For each selected node we also determine (according to the distance to the region of interest) the voxel-to-texel relation.

   The key point of this selection process is to determine how volume data encoded in the HO has to be represented in texture memory to reduce the required texture space [4].

2. The second step visualizes the nodes applying the node visualization function previously described.

5 EXPERIMENTAL RESULTS

Two different tests were performed to evaluate the proposed hybrid visualization function. The first tests were devoted to prove the capabilities of the function to integrate surface and volume data in a single image. In figure 6 we illustrate the transition between two neighbor HO nodes represented at different resolutions (i.e. with different voxel-to-texel ratio). The surface polygons have been positioned at the border of both nodes. It can be seen that no discontinuities are perceptible. Therefore the function guarantees a correct visualization and the continuity of the surface.

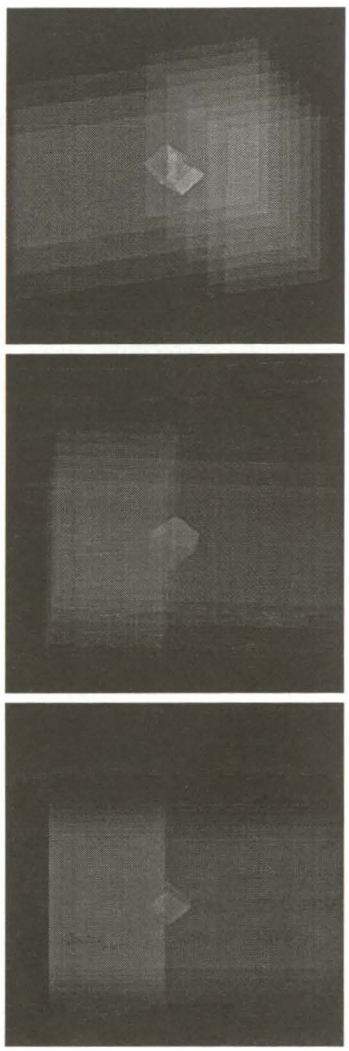

Figure 6. Surface data has been positioned in the border of nodes represented with different accuracy

The second tests were devoted to evaluate the capabilities of performing multiresolution hybrid renderings. In this tests we apply the error-driven adaptive strategy presented in section 4.2.

The images of figure 7 are obtained from the CT-head dataset of 128x128x128. Simulations have been performed on a SGI Octane workstation using a single 270Mz/R12000 processor with 4MB of texture memory.

The size of the texture cache is 128x128x64, thus the size of the maximal texture is 64^3. This CT-head hybrid scene has been obtained integrating the surface of threshold 155 contained in the jaw area. This surface is composed of 9.219 polygons. The volume data has been rendered by adopting different voxel-to-texel ratio, the space required by the function to represent the volume in texture memory is , from up to down, 8, 4 and 2MB. As at each step the texture space required decreases rendering speed is improved.

6 CONCLUSIONS AND FUTURE WORK

We have presented a hybrid visualization function that combines the 3D texture-based volume rendering approach with a surface integration process. The function has been applied for the visualization of an Hybrid Octree, the data structure used to maintain the hybrid scene.

It has been proved that the proposed function is able to perform *multiresolution* hybrid visualizations with the guarantee of an efficient use of texture memory space. Observe that all the surface classification process required for these visualizations depends on the volume data representation. Therefore the application of the multiresolution texture representation reduces the complexity of the surface-volume slab sorting process, a feature that becomes of special interest if we have to deal with large data sets.

Future work will be centered on the definition of new strategies to select the nodes of the Hybrid Octree from which the hybrid scene has to be obtained. Our main goal is to determine the *best* strategy that simplifies the surface data integration process maintaining image quality.

7 ACKNOWLEDGEMENTS

The present work has be partially supported by the Spanish Ministry of Science and Technology grants: TIC2001-2226-C02, 2FD1997-1511

References

[1] Akeley K., Reality Engine Graphics. Computer Graphics (ACM Siggraph Proceedings), 27:109-116,1993.

[2] Boada I., Towards Multiresolution Integrated Surface and Volume Data Representations. PhD Thesis, Universitat Politècnica de Catalunya, September (2001).

[3] Boada, I. and Navazo, I., The Hybrid Octree: Towards a Multiresolution Hybrid Framework. 2002 International Conference on Computer Science. LNCS-series. P.M.A. Sloot, J.J. dongarra, C.J.K. Tan and A.G.Hoekstra (Eds) Springer Verlag, Part II, pp.121-130 April 2002.

[4] Boada, I., Navazo, I. and Scopigno, R., Multiresolution Volume Visualization with a Texture-based Octree. The Visual Computer, Springer International, 17 (3), pp. 185-197, 2001.

[5] Boada, I. and Navazo, I., An Octree Isosurface Codification based on Discrete Planes. Proceedings of Spring Conference on Computer Graphics 2001. pp.187-194. Budmerice,Slovakia.

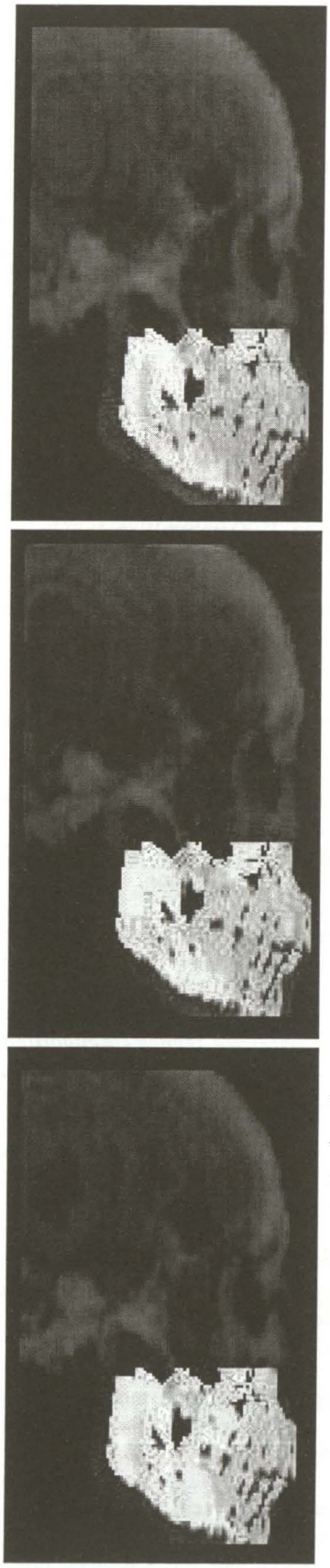

Figure 7. Different renderings obtained on the CT-head. Surface polygons represent the teeth region.

[6] Boada I. and Navazo I., Multiresolution Isosurface Fitting on a Surface Octree. Vision, Modeling and Visualization 2001 proceedings, 318-324, Stuttgart, November(2001).

[7] Cabral B., Cam N. and Foran J., Accelerated Volume Rendering and Tomographic Reconstruction using Texture Mapping Hardware. In ACM Symposium on Volume Visualization,pp. 91- 98 Washington, D.C. October 1994.

[8] Sheng-Yih Guan and Richard Lipes. Innovative Volume rendering using 3D Texture Mapping. In Image Capture, Formatting and Display.SPIE 2164, 1994.

[9] Kaufman, A. and Shimony, E., 3D Scan Conversion Algorithms for Voxel Based Graphics, Proceedings ACM Workshop on Interactive 3D Graphics, Chapel Hill, NC, 45-75, October 1986.

[10] Kaufman, A., Efficient Algorithms for 3D Scan-Conversion of parametric Curves, Surfaces and Volumes, Computer Graphics, 21, 4, 171-79, July 1987.

[11] Kaufman, A., Efficient Algorithms for 3D Scan-Converting Polygons, Computer and Graphics, 12, 2, 213-219,1988.

[12] Kaufman A., Yagel R. and Cohen D., Intermixing Surface and Volume Rendering. 3D Imaging in Medicine, Edited by k,H. Hohne et al. Springer Verlag Berlin Heidelberg 1990

[13] Kreeger K. and Kaufman A., Mixing translucent Polygons with Volumes. IEEE Visualization 99.

[14] David Laur and Pat Hanrahan. *Hierarchical Splatting: A progressive refinement algorithm for volume rendering*. Computer Graphics (ACM Siggraph Proceedings), 25 (4):285-288, July 1991.

[15] Levoy M., A Hybrid Ray Tracer for Rendering Polygon and Volume Data. IEEE Computer Graphics and Applications, 10, 33-40, March 1990.

[16] W.Lorensen and H.Cline, Marching cubes a high resolution 3D surface construction algorithm,. ACM Computer Graphics (Proceedings of SIGGRAPH '87), vol.21, n 4, pp 163-170, 1987.

[17] D.Meagher, Geometric modeling using octree encoding. Computer Graphics and Image Processing, 19(2):129-147,(1982).

[18] C.Montani, R.Scateni and R.Scopigno, Discretized Marching Cubes, in Visualization '94 Proceedings, R.D. Bergeron and A.E.Kaufman, Eds. (1994), 281-287, IEEE Computer Society Press.

[19] C.Montani, R.Scateni and R.Scopigno, Decreasing Isosurface Complexity via Discrete Fitting, Computer Aided Geometric Design, 17 (2000) 207-232.

[20] I.Navazo, Extended Octree Representation of General Solids with Plane Faces: Model Structure and Algorithms. Computer and Graphics, vol 13, 1, (1989), 5-16.

[21] B.A. Payne and A.W. Toga, Distance Field manipulation of surface models. IEEE Computer Graphics and Applications, 12(1), 65-71. January 1992.

[22] H.Samet, Applications of Spatial Data Structures. Addison Wesley, Reading, MA, (1990).

[23] R.Shekhar, E.Fayyad, R.Yagel and J.Cornhill. Octree based Decimation of Marching Cubes surfaces. Visualization 96, 335-342, 1996.

[24] L.M.Sobierajski and A. Kaufman, Volumetric Ray-tracing. In Proceedings of 1994 Symposium on Volume Visualization, pp.11-18. ACM Press, October 1994.

[25] M. Sramek, Non-binary Voxelization for Volume Graphics. In Proceedings of Spring Conference on Computer Graphics, 2001. pp. 35-51.

[26] D.Tost, A.Puig and I.Navazo, Visualization of mixed scenes based on volume and surface. In Proceedings of the Fourth Eurographics Workshop on Rendering, pp.281-294, 1993.

[27] R.Westemann, L.Kobbalt and T.Erl. Real-time exploration of Regular Volume Data by Adaptive Reconstruction of Isosurfaces. The Visual Computing 1998

[28] Orion Wilson, Allen Van Gelder, Jane Wilhems, Direct Volume rendering via 3D textures. Technical Report UCSC-CRL-94-19, University of California, Santa Cruz, June 1994.

[29] J. Wilhems and A. Van Gelder. Octrees for Faster Isosurface generation. ACM Transactions on Graphics, 11(3). 201-227, July 1992.

[30] J.Wilhems and A. Van Gelder. Multi-dimensional Trees for Controlled Volume Rendering and Compression. In proceedings of 1994 Symposium on Volume Visualization, pp 27-34. ACM Press, October 17-18 1994.